Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

ASP.NET MVC a vývoj webových aplikací pro mobilní zařízení

ASP.NET MVC and Web Application Developing for Mobile Devices

2014 Monika Henčeková

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Monika Henčeková**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: ASP.NET MVC a vývoj webových aplikací pro mobilní zařízení
ASP.NET MVC and Web Application Developing for Mobile Devices

Zásady pro vypracování:

Microsoft ASP.NET MVC Framework je technologie, která se v posledních letech velice rozšiřuje a v rámci vývoje ASP.NET aplikací je alternativou k ASP.NET Web Forms. V rámci této bakalářské práce studentka prostuduje ASP.NET MVC 4.0 a na vybraných příkladech bude demonstrovat stěžejní fáze vývoje aplikací s využitím této technologie a zaměřením na výstup pro uživatele mobilních zařízení.

Jednotlivé části práce budou:
1. Prostudovat a popsat technologii ASP.MVC.
2. Prostudovat vývoj aplikací v ASP.MVC s výstupem přizpůsobeným pro mobilní aplikace.
3. Pomocí ASP.MVC implementovat aplikaci, která bude obsahovat prvky, jakými jsou: interaktivní chat, kalendář, jednoduchý seznam produktů, foto galerie apod.
4. Provedení testování a jeho vyhodnocení vytvořené aplikace na různých mobilních platformách.

Seznam doporučené odborné literatury:

MVC : The Official Microsoft ASP.NET Site: http://www.asp.net/mvc

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Martinovič, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA 1

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne. Uviedla som všetky literárne parametre a publikácie, z ktorých som čerpala.

V Ostrave 7.mája 2014 .....

Rada by som na tomto mieste poďakovala vedúcemu mojej baklalárskej práce Ing. Janovi Martinovičovi Ph.D. za odborné vedenie a rady v priebehu práce.

Abstrakt

Táto bakalárska práca popisuje technológiou ASP.NET MVC4 a vývojom webových aplikácií zameraných na mobilné zariadenia pomocou tejto technológie. Aplikáciou som otestovala funkčnosť a implementáciu bežných webových prvkov ako kalendár akcií, galéria a interaktívny chat v mobilnom rozhraní. Práca stručne rozoberá históriu a postupný vývoj frameworku ASP.NET MVC od jeho vytvorenia až po v súčasnosti posledne vydanú verziu. V samostatnej kapitole je rozobraná koncepcia frameworku MVC. Súčasťou práce je aj testovanie ukážkovej aplikácie a následné zhrnutie výsledkov testovania.

Kľúčové slová: ASP.NET MVC4, Model-View-Controller, MVC, Razor, Ajax, jQuery, LINQ, JavaScript, mobilné zariadenia

Abstract

This bachelor thesis describes the technology MVC4 ASP.NET and web development aimed at mobile devices using this technology. By testing application I tested functionality and implementation of common web elements such as event calendar, gallery and interactive chat on a mobile interface by this application. The thesis briefly discusses the history and gradual development of ASP.NET MVC framework since its creation to recently released version. In a separate chapter is discussed concept of MVC framework. Part of this thesis is also testing a sample application and summary of test results

Keywords: ASP.NET MVC4, Model-View-Controller, MVC, Razor, Ajax, jQuery, LINQ, JavaScript, mobile devices

Zoznam použitých skratiek a symbolov

TDD	–	Test driven development (Vývoj riadený testami)
MVC	–	Model View Controller (Architektonický vzor)
HTML	–	Hyper Text Markup Language (Hypertextový značkovací jazyk)
HTTP	–	Hypertext Transfer Protocol (Hypertextový prenosový protokol)
HTTPS	–	Hypertext Transfer Protocol Secure (Zabezpečený hypertextový prenosový protokol)
IIS Express	–	Internet Information Services Express
LINQ	–	Language–Integrated Query
JS	–	JavaScript (interpretovaný skriptovací jazyk)
WAP	–	Wireless Application Protocol (aplikačný protokol pre bezdrôtové zariadenia)
ORM	–	Object-relational mapping (Objektovo relačné mapovanie)
CSS	–	Cascading Style Sheets (Kaskádové štýly)
CPU	–	Central processing unit (Centrálna procesorová jendotka - hlavný procesor počítača)
T-SQL	–	Transact SQL (Tranzakčný SQL)

Obsah

Zoznam tabuliek

Zoznam obrázkov

Zoznam výpisov zdrojového kódu

1 Úvod

Za posledné roky má prehliadanie webu prostredníctvom mobilných zariadení rastúcu tendenciu. Celosvetový podiel prístupu na web z mobilných zariadení dosiahol v máji 2013 15% z celkového počtu prístupov [1]. Je to medziročný nárast až o 5 percentuálnych bodov. Všetko nasvedčuje tomu, že tento trend bude narastať. Preto je na mieste riešiť problematiku webových aplikácií vhodných pre mobilné zariadenia. Používatelia mobilných zariadení sa pri prezeraný webu cez toto zariadenie správajú inak, než pri prezeraní rovnakej stránky na počítači. Pri načítaný stránky bez prispôsobenia pre mobilné zariadenie, teda v plnej veľkosti, je pravdepodobné, že užívateľ začne približovať stred stránky, presne tam, kde sú podstatné informácie. Práve takémuto problému predchádzajú webové aplikácie prispôsobené pre mobilné zariadenia

V súčasnosti sú na výber dva spôsoby implementácie takejto aplikácie [1]. Jednou z nich je tzv. responzívny web. Kedy je celá stránka prispôsobená zobrazovaniu na prehliadačoch PC aj mobilných prehliadačoch, vďaka špecificky určenému CSS. Druhým spôsobom je vytvorenie aplikácie zameranej priamo na mobilné zariadenia. Aj keď je tento spôsob finančne náročnejší než responzívny web, v prípade ak potrebujeme aj verziu pre PC prehliadače. Z technického hľadiska je však táto implementácia oveľa vhodnejší. Aplikácie, ktoré sú vytvorené primárne pre mobilné zariadenia by mali využívať technológie, ktoré urýchľujú načítanie obsahu a zmenšujú celkovú šírku pásma pri požiadavkách.

Z týchto dôvodov som sa rozhodla daný problém preštudovať a následne vytvoriť aplikáciu, na ktorej demonštrujem využívanie technológií zameraných pre efektívne zobrazovanie a správanie sa webových aplikácií na mobilných zariadeniach. Na vytvorenie bola použitá technológia spoločnosti Microsoft, ASP.NET MVC, ktorá je medzi vývojármi stále viac a viac obľúbená.

1.1 Štruktúra práce

Práca sa skladá z niekoľkých častí. Kapitola 2 popisuje postupný vývoj technológie ASP.NET MVC. Každej verzií frameworku MVC je venovaná samostatná časť kapitoly. Po oboznámení sa s vývojom frameworku a jeho štruktúrou v časti 2.2, nasleduje kapitola 3, kde je popísaná funkčnosť a práca s ukážkovou aplikáciou. Kapitola 4 stručne popisuje vývoj mobilného webu a jednotlivé technológie, ktoré boli kľúčové pri postupnom vývoji mobilného webu do podoby, ako ho poznáme dnes. V kapitole 5 je postupne detailne rozobraná štruktúra každého z troch hlavných prvkov architektonického vzoru ASP.NET MVC 4. Kapitola 6 popisuje technológiu Ajax a jednu z najrozšírenejších JavaScriptových knižníc jQuery. Kapitola 7 je venovaná testovaniu ukážkovej aplikácie. Testovanie prebiehalo dvoma spôsobmi. Prvým bolo testovanie reálnym užívateľom na fyzickom zariadení v časti 7.4 a pomocou simulovaných testov v časti 7.5.1.

2 Technológia ASP.NET MVC

ASP.NET MVC [2] je open source webový vývojový framework založený na Microsoft .NET platforme. Poskytuje možnosť vyvíjať dobre štruktúrované webové aplikácie. Základnou myšlienkou MVC je rozdelenie aplikácie do logických celkov modelu (dátová časť), pohľadu (prezentačná časť) a kontrolóra (riadiaca časť), ktoré vedú vývojára cestou návrhového vzoru k jednoduchšiemu, prehľadnejšiemu a robustnejšiemu návrhu celej webovej aplikácie.

2.1 Vývoj ASP.NET MVC

Jedná sa o pomerne mladú technológiu. Prvé verejné predstavenie ASP.NET MVC bolo v novembri roku 2007. Technológia prechádza postupným vývojom a doplňovaním o mnohé vylepšenia a prvky, ktoré uľahčujú prácu s MVC frameworkom a zároveň umožňujú vytvárať čistejší a prehľadnejší kód. Nasledujúce kapitoly popisujú postupný vývoj spolu s hlavnými novinkami, ktoré sa v jednotlivých verziách objavili.

2.1.1 ASP.NET MVC 1.0

Prvá oficiálna verzia tohto frameworku bola vydaná v marci 2009. Vývojovým prostredím sa stalo Visual Studio 2008. ASP.NET MVC 1.0 je postavený na platforme .NET 3.5 a ako zobrazovací nástroj pre pohľady využíva WebForm zobrazovací nástroj (ASPX). Celá koncepcia frameworku bola od začiatku navrhnutá tak, aby umožňoval prehľadné a rýchle písanie kódu a oddelenie dátovej vrstvy od kontrolóra a pohľadu. K tomu pomáhajú aj HTML helpers metódy, ktoré zjednodušujú zápis HTML značiek, viac o HTML helpers v časti 5.4.4. Ajax helpers slúžia na skrátenie a zjednodušenie zápisu funkcií využívajúcich jQuery a JavaScript knižnice (popís v kapitole 6). Navigáciu medzi jednotlivými pohľadmi MVC aplikácií zaisťuje tzv. Routing. Routing je jednou z najdôležitejších súčastí ASP.NET MVC aplikácií. Jeho prácou je mapovanie prichádzajúcich požiadaviek od prehliadača na jednotlivé akcie kontrolóra. MVC je vyvíjané s priamou podporou myšlienky TDD, tedy vývoja riadeného testami. K tomu slúžia vo frameworku Unit testy. Vďaka týmto testom je možné overiť, či kontrolór vrátil správny pohľad spolu s jeho dátami. Taktiež umožňujú testovať presmerovania na iné kontrolóry a pohľady.

2.1.2 ASP.NET MVC 2

Oficiálne vydanie sa datuje na 10. marca 2010. Preberá všetky prvky od MVC 1.0. Platfourmou pre túto verziu sa stáva .NET 4.0 a vývoj je možný vo Visual Studiu 2010. Ako jedna z noviniek sa v druhom vydaní ASP.NET MVC objavili silne typové HTML helpers metódy založené na lambda výrazoch. Tieto metódy umožnili lepší čas kompilovania. Templated helpers [3] poskytujú nástroje na automatické zostavenie užívateľského rozhrania založeného na dátach modelu. Na validáciu dát poskytuje MVC 2 Data annotations attributes, ktoré umožňujú každému atribútu pridať ďalší popis a rozsah kontextu parametru, vďaka čomu je potom možné generovať chytrejšiu validáciu. Po pridaní atribútov

do modelu je podporovaná validáca na klientskej aj serverovej strane, viac o Data annotations attributes v časti 5.2.2. Ďalšou podstatnou novinkou, ktorá prišla v tejto verzií je možnosť rozdelenia veľkých aplikácií do oblastí. Celá aplikácia je tak rozdelená do malých funkčných skupín. Aplikácia môže obsahovať niekoľko oblastí. Zabezpečí sa tým dobrá prehľadnosť jednotlivých tried. Poslednou významnou novinkou ASP.NET MVC 2 sú Asynchronous Controllers (asynchrónne kontrolóry). Tieto kotrolóry umožňujú spracovávať požiadavky asynchrónne. Využiteľné sú napríklad pri dlho bežiacich metódach, ktoré nie sú viazané na CPU. Typickým príkladom použitia asynchrónneho kontrolóra je pri dlhotrvajúcom hovore cez webovú službu.

2.1.3 ASP.NET MVC 3

Vydaný 13. januára 2011. Pracuje výhradne na platforme .NET 4 a novšej. Hlavná zmena oproti predošlým verziám je v zobrazovacom nástroji. Okrem doterajšieho WebForms zobrazovacieho nástroja pribudol nový zobrazovací nástroj Razor. Na strane Ajaxu a validation helpers [4] je primárne využívaný prístup unobtrusive JavaScriptu, to znamená, že JS nie je písaný priamo medzi hmtl značky (ako inline kód), ale je oddelený od zvyšku kódu. Využívanie unobtrusive (nevtieravého) JavaSriptu [5] predchádza písaniu priamo medzi HTML značky, miesto toho je využitá separácia správania, využívajúca konvencie HTML5. Pribudla možnosť vzdialenej validácie atribútov, ktorá priniesla výhody jQuery Validation pluginov. Tým je umožnené knižniciam s validáciou na klientskej strane automaticky volať užívateľské metódy, ktoré sú definované na servery, aby tak bola vykonaná validačná logika, ktorý je prístupná len na strane serveru. Ako ďalšia novinka je možnosť vytvárania kontrolórov bez sessions. Taktiež je možné rozhodnúť či daný kontrolór bude na čítanie/zápis alebo len na čítanie. Po prvýkrát sa objavila možnosť využitia Entity frameworku podporujúceho techniky *Code First, Model fisrt* a *Database first*, viac o Entity frameworku v časti 5.2.1. Ďalej pribudla možnosť čiastočne ukladať stránku do vyrovnávacej pamäte. To znamená, že nie je potrebné ukladať celý pohľad a jeho dáta, ale iba vybrané oblasti či fragmenty. Doterajšie vlastnosti Controller.ViewModel a View, ktoré zabezpečovali prechádzanie dát medzi kontrolórom a pohľadom, boli premenované na zjednodušený a jednotný názov ViewBag, podrobnejšie informácie 5.4.6.

2.1.4 ASP.NET MVC 4

Uvoľnenie plnej verzie bolo 15. augusta 2012. Pribudla možnosť využitia novej verzie .NET frameworku vo verzií 4.5 a vývojového prostredia Visual Studio 2012. Významnou novinkou je možnosť v rámci MVC aplikácie využívať Web API rozhranie na vytváranie a využívanie HTTP služieb so širokou ponukou klientov, prehliadačov, mobilov a tabletov. Toto rozhranie je vhodné hlavne pri vytváraní služieb dodržujúcich štýl architektúry REST. Nový vzhľad štandardnej šablóny pre pohľady. Vykresľovanie je pomocou techniky nazvanej adaptívne renderovanie, vďaka čomu je zabezpečený príslušný vzhľad pre všetky zariadenie, ako pre počítače tak aj pre mobilné zariadenia. Šablóna na mobilné aplikácie využívajúca jQuery Mobile, prináša možnosť vytvorenia aplikácie špecializovanej na mobilné zariadenia bez potreby dodatočných úprav a špecifikácií. Zobrazovacie

módy dovoľujú aplikácií zvoliť pohľad, na základe prehliadača, ktorý podal požiadavku. Je to výhodné hlavne pri mobilných zariadeniach, kedy nie je potreba vykresľovať „plný" pohľad ako je to pri klasických počítačoch, kde užívateľ očakáva plné grafické zobrazenie.

2.1.5 ASP.NET MVC 5

Uvoľnenie plnej verzie pre verejnosť bolo 17. októbra 2013. Podporovanou platformou je .NET verzie 4.5 a novšia 4.5.1. Vývoj je možný len vo vývojovom prostredí Visual Studiu 2013 a vo Visual Studiu 2012 po doinštalovaní balíčka obsahujúceho webové nástroje ASP.NET. V novej verzií MVC 5 je možné využívať Entity Frameworku vo verzií 6. Asi najvýraznejšou novinkou, ktorú si je možné všimnúť hneď pri vytváraní projektu je One ASP.NET. Ide o sprievodcu pri vytváraní aplikácie, kde je možné vybrať šablónu projektu a nadstaviť spôsob autentifikácie. Na autentifikáciu sa v novej verzií ASP.NET MVC používa ASP.NET Identity a manažment identity. To prinieslo možnosť prihlasovania sa do aplikácií pomocou účtu na Facebooku alebo Google účtu. Úplnou novinkou sú autentifikačné filtre, ktoré umožňujú autentifikačnú logiku na akciu, kontrolór alebo globálne na všetky kontrolóry. Pomocou autentifikačných kontrolórov je taktiež možné pridávať výzvy na autentifikáciu, pri vykonaní neoprávneného požiadavku. Filtre je možné preťažovať a tým špecifikovať, ktorý sa použije pri danej metóde alebo kontrolórovy. Je tak možné globálne nakonfigurované filtre preťažiť a vylúčiť určité filtre a ich funkcie z akcií alebo kontrolórov. Zmenou prešla aj štandardná vzhľadový šablóna. Bootstrap vzhľad je známy hlavne zo sociálnej siete Twitter. Bootsrap framework [6] je plne prispôsobiteľný, responzívny a ľahko nesaditeľný. Špecifikáciu smerovania pomocou anotácií na kontrolóry a akcie umožňuje Attribute routering. Pôvodne balíček pridávaný pomocou správcu balíčkov Nugget. V novej verzií MVC 5 je už plne integrovaný do projektov.

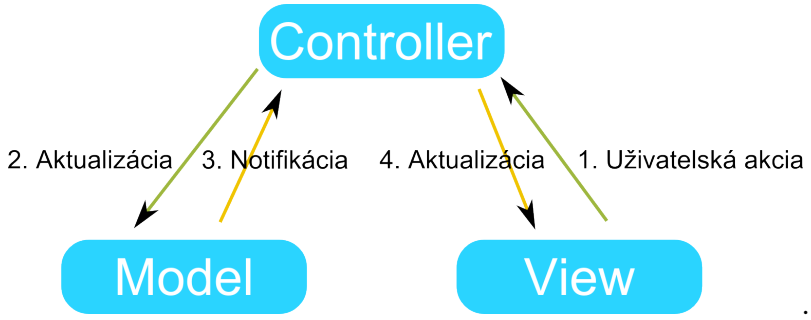
2.1.6 ASP.NET MVC 5.1

20. januára 2014 Microsoft oficiálne vydal plnú verziu APS.NET MVC 5.1. Je postavený na platforme .NET 4.5.1. Nejde tak ani o výlučne novú verziu, než skôr o vylepšenie verzie 5. Noviniek je pomenej, než pri vydaní predošlých verzií. Jednou z mála noviniek je podpora enum typov priamo v pohľade. S enum typmi sa pracuje ako so zoznamami. Na prácu s nimi slúžia HTML helpers. Pomocnými metódami je možné napríklad zobraziť hodnoty enumu ako rozbaľovací zoznam. Ďalšou zmenou je upravenie používania HTML pomocnej metódy EditFor(), ktorý v novej verzií MVC 5.1. môže využívať odvodené HTML attributes. Umožňuje prechádzať html attributes ako anonymné objekty. Zmenou prešla aj validácia vstupov na klientskej strane. Reťazcové vstupy a typy polí môžu byť ľahko validované pomocou vlastností MinLengthAttribute a MaxLengthAttribut, ktoré overujú zadanú dĺžku na základe jej definovania v modely. Novinkou je aj možnosť pristupovať a vyvolať funkcie spätného volania pomocou kľúčového slova this.

2.2 Alternatíva k Web Forms

ASP.NET MVC framework ponúka alternatívu k ASP.NET Web Froms na vytváranie webových aplikácií založených na trojvrstvovej architektúre. ASP.NET MVC poskytuje výrazný posun vpred so zameraním na čistotu kódu, oddelenie jednotlivých vrstiev a testovateľnosť.

Architektonický vzor [7] rozkladá aplikáciu do troch hlavných komponentov: model, view a controller (obrázok 2.1). Model obsahuje dátovú vrstvu a business logiku, predstavuje jadro celej aplikácie. View je komponent, ktorý slúži na zobrazovanie užívateľského rozhrania. Typicky sa toto rozhranie vytvára z modelových dát. Controller slúži na zachytávanie interakcie užívateľa, pracuje s modelom a rozhoduje, ktoré pohľady (view) na základe interakcií užívateľa zobrazí. Aplikácie postavené na ASP.NET MVC je možné vytvorit' pomocou jedného z troch jazykov: C#, Visual Basic a F#.



Obrázok 2.1: Model-View-Controller

2.2.1 Podpora viacerých pohľadov v jednej aplikácií

Jednou z hlavných výhod MVC frameworku je podpora viacerých pohľadov. Vďaka tomu, že model je plne oddelený od pohľadu a nie je žiadna priama závislost' modelu na pohľade, môže byt' užívateľské rozhranie zobrazené viacerými pohľadmi s rovnakými dátami v rovnaký čas. V praxi to znamená využívanie objektov jedného modelu viacerými stránkami. Ďalším príkladom prínosu tejto separácie je vytvorenie webovej aplikácie, ktorá umožňuje užívateľovi zmenit' vzhľad stránok. Tieto stránky zobrazujú rovnaké dáta zo zdieľaného modelu, ale zobrazujú ich rôznymi spôsobmi. Rôzny užívatelia preferujú rozdielne farby, štýly písma, rozloženia stránky a podporu nových zariadení ako mobilné telefóny a tablety. Keďže je model plne nezávislý na pohľade, pridávanie nových typov pohľadov do aplikácie nemá žiaden vplyv na model. V dôsledku toho sa zmeny prejavia len v zobrazení samotnej aplikácie.

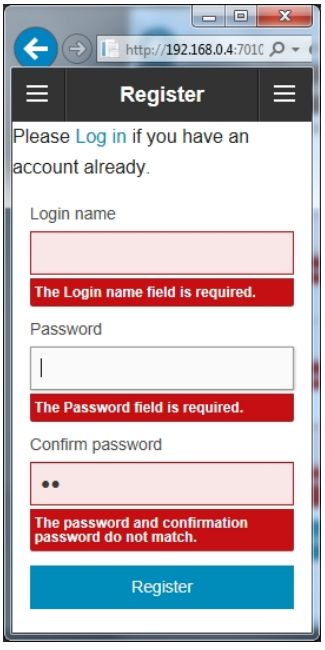
3 Ukážková aplikácia

Ukážková aplikácia, ktorá je súčasťou tejto práce, je vytvorená pomocou technológie ASP.NET MVC 4. Je špecializovaná pre mobilné zariadenia ako sú chytré mobilné telefóny a tablety. Aplikáciou a celou bakalárskou prácou chcem poukázať na potrebu vytvárania mobilných verzií webových aplikácií. Nakoľko takto vytvorené aplikácie sú pre mobilné zariadenia vhodnejšie než responzívny web. Aplikácia obsahuje galériu, interaktívny chat, zoznam nadchádzajúcich udalostí a užívateľ má taktiež možnosť ku konkrétnej udalosti si stiahnuť súbor s kalendárom, ktorý mu pridá túto udalosť do kalendára zariadenia.

3.1 Register

Aplikácia je rozdelená do častí, ktoré sú prístupné bez registrácie a do častí, ktoré sú prístupné len po zaregistrovaní a následnom prihlásení sa do aplikácie. Na registrovanie nového užívateľa je vytvorená samostatná stránka Register, kde užívateľ zadá svoju prezývku Nick, pod ktorým chce vystupovať v chate a heslo. Heslo je potrebné potvrdiť opätovným zadaním do kolonky Confirm Password, aby sa overilo, že užívateľ vie aké heslo zadal.

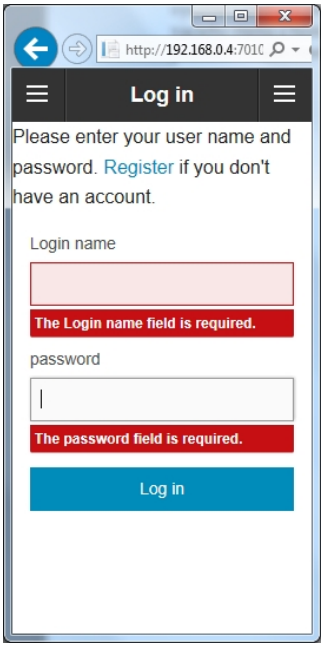
Pri registrácií sú využívané validátory, ktoré sú podrobne popísané v kapitole 5.2.2. Validátory umožňujú kontrolu prichádzajúcich dát oproti modelu. Na obrázku 3.1 je možné vidieť stránku s registračným formulárom a zobrazenie upozornení, zabezpečených validáciou dát pomocou validátorov.



Obrázok 3.1: Stránka Register

3.2 Log in

V prípade, že užívateľ už má vytvorenú registráciu, môže sa pomocou svojho prihlasovacieho mena (Nick) prihlásiť do aplikácie na stránke Login. Na stránku sa užívateľ dostane jedným kliknutím na ľavé horné menu a následne na Log in. Užívateľovi je zobrazená stránka s prihlasovacím formulárom, kde zadá prihlasovacie meno a heslo. Pri prihlásení užívateľa príde k zaslaniu autentifikačnej cookie [8] do prehliadača, pomocou ktorej následne aplikácia kontroluje či je užívateľ stále prihlásený a pod akým prihlasovacím menom je prihlásený. Ak užívateľ nemá ešte vytvorenú registráciu, ale zobrazil prihlasovaciu stránku, má možnosť kliknutím na odkaz Register v hornej časti stránky zobrazit registračnú stránku. Opäť sú tu využité validácie vstupov pomocou validátorov. Tentokrát je overovaná zhoda prihlasovacieho mena a hesla voči dátam v databázy. Pokiaľ zadá užívateľ nesprávnu kombináciu prihlasovacieho mena a hesla zobrazí za mu upozornenie ako na obrázku 3.2.



Obrázok 3.2: Stránka Log in

3.3 Log off

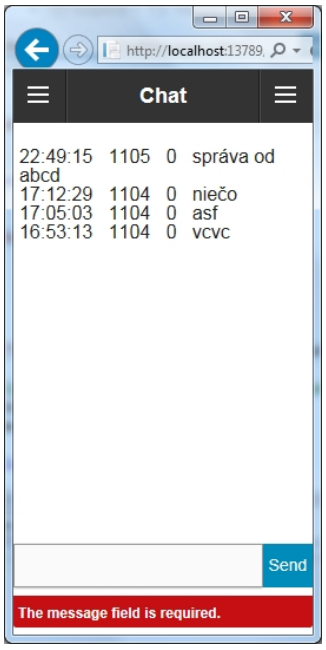
Ak sa užívateľ rozhodne, že v aplikácií nechce už byť prihlásený a využívať časti aplikácie pre prihláseného užívateľa, môže sa odhlásiť. Túto možnosť nájde na stránke Log off. Po odhlásení mu bude zamietnutý prístup na stránku Chat a taktiež nebude mať možnosť písať do chatu správy. Odhlásením je zároveň zrušená platnosť autentifikačnej cookie.

3.4 Home

Domovská stránka uchováva prostredníctvom čiastočného pohľadu (Partial view) [9] zoznam nadchádzajúcich udalostí. Každý popis udalosti je zároveň odkaz na stránku s detailom o udalosti. Navigácia v celej aplikácií je pomocou vysúvacieho menu v ľavom hornom rohu, reprezentovaného ikonou troch vodorovných bielych čiar. V menu je zoznam všetkých základných stránok, aj tých, ktoré je možné zobraziť až po prihlásení. Ak užívateľ nie je prihlásený a klikne na odkaz vedúci na takúto stránku, bude automaticky presmerovaný na prihlasovací formulár.

3.5 Chat

V menu aplikácie je pod názvom Chat odkaz vedúci na stránku s chatom. Na tejto stránke má užívateľ možnosť po prihlásení interaktívne a v reálnom čase komunikovať s ostatnými prihlásenými užívateľmi. Najnovšie správy sú zobrazované vo vrchnej časti stránky. Text správy sa zadáva v spodnej časti stránky do textového poľa a odosiela sa kliknutím na tlačítko Send. Užívateľ nemôže odoslať prázdnu správu. Text správy musí mať dĺžku minimálne jeden znak. Na overenie dĺžky správy sú použité v modely už spomenuté validátory. Na obrázku 3.3 je vidieť správu vyvolanú pri pokuse odoslať správu bez textu.

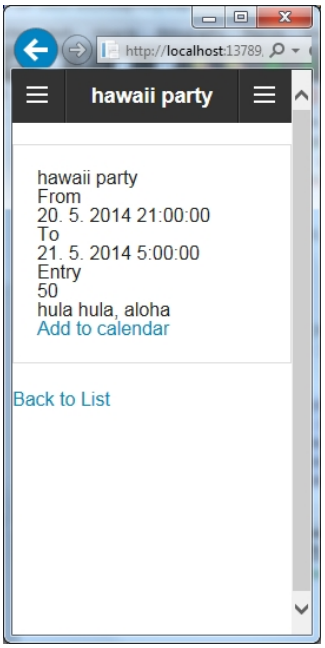


Obrázok 3.3: Stránka Chat

V pohľade stránky Chat je využitý čiastočný pohľad, ktorý zobrazuje zoznam správ a na jeho obnovovanie je využitá technológia Ajax, ktorá je popísaná v kapitole 6.1.

3.6 Events

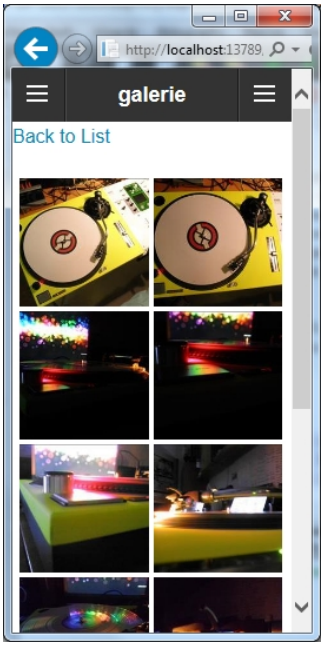
Po kliknutí na odkaz stránky Events v menu aplikácie, je užívateľ presmerovaný na zoznam nadchádzajúcich udalostí. Jedná sa o rovnaký zoznam ako na domovskej stránke. Taktiež je zobrazený pomocou čiastočného pohľadu. Po kliknutí na konkrétnu udalosť, je užívateľ presmerovaný na detail udalosti. V detaile vidí podrobný popis udalosti spolu s časom a dátumom konca udalosti a výškou vstupného. Na tejto stránke má užívateľ možnosť stiahnuť si súbor s kalendárom akcie, ktorý pridá záznam do kalendára jeho zariadenia. Táto možnosť je dostupná na platformách iOS, Android a Windows Phone. Na obrázku 3.4 je vidieť detail konkrétnej udalosti.



Obrázok 3.4: Stránka detailu udalosti

3.7 Galleries

Na stránke Galleries je užívateľovi zobrazený zoznam foto galérií. Užívateľ má možnosť po kliknutí na názov galérie zobrazit v nej obsiahnuté fotky. Po načítaní konkrétnej galérie sú zobrazené fotografie v zmenšenej podobe. Po kliknutí na fotku sa táto zobrazí v pôvodnom rozlíšení a pomocou CSS sa prispôsobí zobrazovacím možnostiam aktuálneho zariadenia užívateľa. Vzhľad galérie so zmenšenými fotkami je vidieť na obrázku 3.5.



Obrázok 3.5: Stránka s fotkami galérie

3.8 Administrátorské práva

Administrátor (admin) aplikácie má rozšírené možnosti oproti bežnému užívateľovi. Po prihlásení sa na svoj účet, môže okrem aktivít prístupných bežnému užívateľovi, vykonávať administráciu jednotlivých častí. Má možnosť, vytvárať, upravovať a mazať galérie, pridávať do galérií fotky, mazať fotky z galérií,taktiež môže vytvárať, mazať a upravovať udalosti. Administrátorské prístupové práva umožňujú aj mazanie správ, v prípade, že by mali nevhodný obsah.

Vývoj ukážkovej aplikácie je popísaný v kapitole 5. V tejto kapitole je každý výpis zdrojového kódu z ukážkovej aplikácie. V kapitole sú postupne popísané stavebné časti (Model, View, Controller) ASP.NET MVC 4 frameworku. V popise každej časti sú následne popísané jej najdôležitejšie súčasti.

K zobrazovaniu webových aplikácií na mobilných zariadeniach je potrebný mobilný web. Jeho stručný vývoj je popísaný v nasledujúcej kapitole.

4 História mobilného webu

Digitálne mobilné technológie bolo možné využívať až po príchode siete druhej generácie (2G). Jej príchodom sa značne rozšírili možnosti využitia mobilných sietí aj v spôsobe prenosu dát [9]. Čo umožnilo postupné sprístupnenie mobilného webu pre širokú verejnosť. K tomu rozšíreniu významne prispeli technológie popísané v nasledujúcich častiach.

4.1 I-Mode

Prvé komerčné spustenie webovej služby postavenej na prehliadaní v internetovom prehliadači a špecifikovanej pre mobilné telefóny bolo v roku 1999 v Japonsku. V tomto roku spustila japonská firma NTT DoCoMo službu pre mobilný internet s názvom „I-Mode“. I-Mode využíva zjednodušenú formu HTML, Compact Wireless Markup Language (CWML) namiesto Wireless Markup Language (WML) využívaného vo WAP

4.2 Nokia 9000

Nokia 9000 z roku 1996 bola vôbec prvým mobilným zariadením schopným prehliadať internet, prijímať e-maily a posielať faxy. Na zobrazovanie vráteného obsahu na displeji mobilného telefónu sa používal TTML (Tagged Text Markup Language). TTML bol značkovací jazyk, ktorý umožňoval optimalizáciu prenosu webových stránok pre mobilné zariadenia, odstránením nepotrebného obsahu ako napríklad reklamy. V rovnaký čas vyvíjala spoločnosť Ericsson ITTP (Intelligent Terminal Transfer Protocol).

4.3 WAP 1.0

Hlavným výrobcom mobilných telefónov bolo jasné, že nejednotnosť trhu nie je dobrá. Rozhodli sa vyvinúť jednotný štandard k zobrazovaniu obsahu internetu na ich mobilných zariadeniach. V júny 1997 Nokia, Ericsson, Motorola a Unwired Planet verejne oznámili spoluprácu na WAP (Wireless Application Protocol). WAP 1.0 mal byť otvorený protokol, ktorý mohol implementovať každý predajca. Tento nový protokol mal zabezpečiť výrobcom mobilných zariadení pripojenie k internetu postavenom na protokole IP, na zariadeniach, ktoré mali vysoký podiel dátovej straty počas komunikácie. Nový protokol so sebou priniesol aj nový štandard návrhu aplikácií, ktoré podporoval.

4.4 Wireless Markup Language

Novým štandardom vo vývoji aplikácií sa stal WML (Wireless Markup Language), značkovací jazyk založený na technológiách HTML a XML. Avšak aj napriek snahe spopularizovať tieto technológie na poli mobilného webu, bol zreteľný problém v ich návrhu. Oba, protokol aj značkovací jazyk, boli totiž navrhnuté pre veľmi pomalé dátové siete a mali veľmi obmedzené zobrazovacie možnosti. S expanziou vývoja mobilných telefónov a PDA boli potrebné nové štandardy, na podporu rozšírenia využívania webu v mobilných zariadeniach. Bol požadovaný nový značkovací jazyk, ktorý by podporoval nové prehliadače, ktoré boli súčasťou mobilných zariadení.

4.5 eXtensible HyperText Markup Language Mobile Profile

Tomuto požiadavku vyhovel v roku 2001 XHTML MP (eXtensible HyperText Markup Language Mobile Profile). Základom pre tento značkovací jazyk sa stal XHTML Basic. XHTML MP nahradil WML v protokole WAP. Ani tento nový štandard nebol veľmi obľúbený. Dnešné mobilné zariadenia majú prehliadače podporujúce HTML štandardy vrátane HTML5 a CSS3.

5 Špecializácia ASP.NET MVC 4 na mobilné zariadenia

Vydaním štvrtej verzie ASP.NET MVC v máji 2012 poskytol Microsoft vývojárom možnosť vyvíjať webové aplikácie optimalizované pre mobilné zariadenia bez potreby špeciálnej konfigurácie. Samozrejmosťou je využívanie najnovších technológií ako HTML5 a CSS3, schopnosť spojiť kompresiu a prenos JavaScrip a CSS súborov k minimalizácií šírky pásma a žiadostí prehliadača. Vývoj webových aplikácií špeciálne navrhnutých pre mobilné zariadenia je dôležitý hlavne z dôvodu, že mobilný prehliadač sa správa inak než prehliadač na PC.Pohľady tak môžu byť vytvárané so zameraním na špecifické prehliadače.

Ďalším dôležitým aspektom mobilných prehliadačov je viewport. Jedná sa o virtuálne okno prehliadača, definované vo väčšine mobilných prehliadačov. Vďaka tomu, že viewport je definovaný s väčšími rozmermi než sú skutočné rozmery obrazovky daného zariadenia, umožňuje zväčšovať obsah zobrazovanej stránky. V prípade ak sú hodnoty viewport nastavené na aktuálne rozmery obrazovky zariadenia, je približovanie obsahu zobrazovanej stránky znemožnené, lebo obsah je nastavený tak, aby pasoval pre danú obrazovku. Nastavenie vlastností viewport je možné pomocou značky <meta/> v hlavičke HTML súboru. Táto značka hovorí mobilnému prehliadaču, aby nastavil šírku viewport na aktuálnu šírku obrazovky zariadenia [9]. Implementácia je uvedená vo výpise 1.

```
<meta name="viewport" content="width=device-width"/>
```

Výpis 1: Použitie viewport v HTML

Toto nastavenie je dosiahnuté pomocou konštanty device-width, ktorej hodnota je definovaná v každom zariadení so šírkou obrazovky. Toto nastavenie je dôležité, preto že užívatelia chcú, aby mobilné verzie webových aplikácií boli vhodné pre ich mobilné zariadenia, nie len pre plné desktopové zobrazenie. Media query je definícia typu média a nastavenie jeho charakteristík. Nasledujúci media query špecifikuje, že daná skupina kaskádových štýlov bude zobrazená len v prípade, že je stránka zobrazená na obrazovke. Čo znamená, že ak budeme chcieť stránku vytlačiť, jej kaskádové štýly nebudú vytlačené s obsahom stránky. Charakteristiky média taktiež udávajú, že maximálna šírka je 40em (relatívna jednotka) ako je vidieť z výpisu 2. Pre prehliadač to znamená, že ak je obrazovka široká 40em alebo viac aplikujú sa dané štýly.

```
@media only screen and (max-width: 40em)
{
    // skupina stylov, ktore aplikuje media query
}
```

Výpis 2: Nastavenie media query v HTML

Všetky výpisy zdrojových kódov v práci sú z ukážkovej aplikácie.

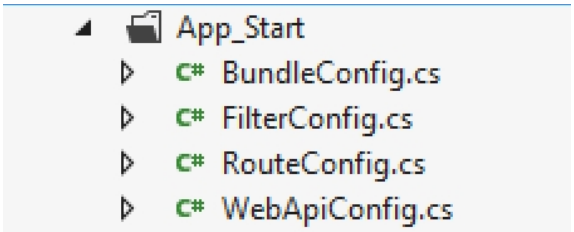
5.1 Global.asax

Súbor Global.asax [9] známy aj ako aplikačný súbor ASP.NET, je voliteľný súbor obsahujúci kód odpovedajúci na udalosti aplikačnej úrovne a session úrovne vyvolané ASP.NET alebo http modulmi. Tento súbor je umiestnený v koreňovom adresári ASP.NET aplikácie. Pri behu je Global.asax parsovaný a kompilovaný do dynamicky generovanej .NET Framework triedy odvodenej z bázovej triedy HttpApplication. Akákoľvek priama URL požiadavka na Global.asax je automaticky odmietnutá. Externý užívatelia nemôžu vidiet' kód v ňom obsiahnutý.

Dôležité udalosti, ktoré sa môžu vyskytovat' v Global.asax:

- **Application_Init** - táto udalost' je vyvolaná, keď je aplikácia inicializovaná po prvýkrát
- **Application_Start** – udalost' je vyvolaná pri prvom spustení aplikácie
- **Application_BeginRequest** – udalost' sa vyvolá pri príchode každej novej požiadavky
- **Application_EndRequest** – udalost' je vyvolaná pri ukončovaní aplikácie
- **Application_AuthenticateRequest** – udalost' indikuje, že požiadavka je pripravená na autentifikáciu
- **Application_Error** – udalost' je vyvolaná ak sa v aplikácií objaví neošetrená chyba
- **Session_End** – táto udalost' je vyvolaná kedykoľvek sa ukončí jedna užívateľská seansa alebo vyprší jej čas
- **Application_End** – udalost' je vyvolaná keď aplikácia končí alebo vypršal jej čas, typicky obsahuje čistiacu logiku

Pri vytvorení aplikácie ASP.NET MVC 4 obsahuje súbor Global.asax východiskovo len jednu metódu Application_Start. Táto metóda pri prvom spustení aplikácie volá všetky naviazané metódy z tried, ktoré sú v zložke App_Start na obrázku 5.1.

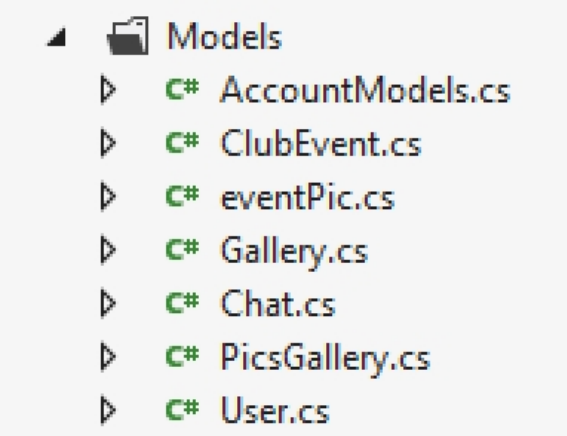


Obrázok 5.1: Zložka App_start

V nasledujúcich kapitolách sú podrobne popísané tri hlavné stavebné piliere architektúry MVC.

5.2 Model

Myšlienka ASP.NET MVC spočíva v prehľadnosti kódu. Túto prehľadnosť máme zabezpečenú umiestňovaním jednotlivých stavebných prvkov – tried aplikácie do príslušných zložiek. Základným kameňom MVC je model. Každá trieda, ktorá je model je vždy umiestnená v zložke s názvom Models, ktorá je vyobrazená na obrázku 5.2



Obrázok 5.2: Zložka Models

Model je trieda, reprezentujúca dáta v databázy. Každá inštancia modelu je jeden riadok v databázovej tabuľke a každá vlastnosť objektu danej triedy je mapovaná na jeden stĺpec danej databázovej tabuľky. Model obsahuje celú aplikačnú logiku a zabezpečuje manipuláciu s dátami v databázy. Každá tabuľka v databázy je reprezentovaná jednou triedou v zložke Models.

Vo výpise 3 je trieda Gallery reprezentujúca tabuľku v databázy s rovnakým názvom. Kľúčové slovo [Key] v začiatku definície triedy určuje, že vlastnosť galleryID je v databázy primárnym kľúčom. Nasledujú vlastnosti reprezentujúce jednotlivé stĺpce tabuľky. Posledná vlastnosť udržiava kolekciu triedy PicsGallery, čo je asociačná trieda, reprezentujúca asociačnú tabuľku v databázy.

```
public class Gallery
    {
        [Key]
        public int galleryID { get; set; }
        public DateTime addDate { get; set; }
        public string galleryTitle { get; set; }
        public virtual ICollection<PicsGallery> PicsGallery { get; set; }
    }
```

Výpis 3: Modelová trieda Gallery

5.2.1 Entity Framework

V aplikácií je na prístup k dátam využitá technológia známa ako Entity Framework [**?**] (často označovaná jednoducho ako EF). Jedná sa o flexibilný ORM nástroj, ktorý umožňuje pracovať s databázovými dátami objektovo orientovaným spôsobom.

Podporuje vývojárske prístupy nazvané *Code first, Database first a Model first*. Pomocou *Code first* prístupu je možné ako prvé vytvoriť modelové objekty napísaním jednoduchých tried – modelov. Z týchto tried je potom možné vytvoriť databázu priamo za behu aplikácie.

Funkcionalitu Entity Frameworku pre daný model dát koordinuje databázová kontextová trieda GlobalDBContext. Táto trieda je odvodená z menného priestoru System.Data.Entity.DBContext. Databázová kontextová trieda umožňuje prispôsobovať správanie Entity Frameworku. Vďaka čomu je vývoj rýchly a čistý. Nasledovný kód vytvorí vlastnosť DbSet pre každý entity set. Termín entity set v Entity Frameworku typicky korešponduje s databázovou tabuľkou a entita odpovedá riadku v tabuľke. Táto vlastnosť sa správa ako kolekcia dovoľujúca dopytovať sa na dáta v databázových tabuľkách, ako keby boli kolekciami objektov v pamäti. DbSet je typovaný entity set, ktorý je použitý na vykonávanie CRUD operácií. Entity Framework vygeneruje odpovedajúci SQL príkaz na tabuľku a konvertuje výsledok príkazu do silne typového objektu. DbSet je možné vytvoriť len z inštancie DbContext. Metóda OnModelCreating() zabraňuje pluralizácií názvu tabuliek počas vytvárania databázy.

```
public class GlobalDBContext : DbContext
    {
      public DbSet<Gallery> GalleriesSet { get; set; }
      public DbSet<eventPic> PicSet { get; set; }
      public DbSet<ClubEvent> ClubEventsSet { get; set; }
      public DbSet<PicsGallery> PicsGallerySet { get; set; }

      protected override void OnModelCreating(DbModelBuilder modelBuilder)
            {
                modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();
            }
    }
```

Výpis 4: Trieda GlobalDBContext

Hodnoty, ktoré sú prostredníctvom vlastností modelu vkladané do databázy vytvorenej pomocou Entity frameworku, je potrebné nejakým spôsobom kontrolovať. Spôsoby kontroly týchto hodnôt sú popísané v nasledujúcej kapitole.

5.2.2 Validácia užívateľských vstupov v modely

Validácia užívateľský vstupov začína už na úrovni modelu. Kontrola vkladaných záznamov je dôležitá z hľadiska ukladania dát do databázy v správnom formáte či dĺžke. Validácie je možné jednoducho pridávať pomocou atribútov [10] obsiahnutých v triedach, ktoré sa nachádzajú v mennom priestore System.ComponentModel.DataAnnotations.

Medzi najčastejšie používané validačné atribúty patria:

- **Required** – určuje, že daná vlastnosť je povinná, preto vstupné pole nemôže zostať nevyplnené
- **DisplayName** – definuje názov, ktorý bude použitý na poly formuláru a vo validačných správach
- **StringLength** – umožňuje definovať maximálnu možnú dĺžku reťazca
- **Range** – určuje minimálnu a maximálnu hodnotu číselných vstupov
- **ScaffoldColumn** – umožňuje skrývať polia v editačných formulároch

Po pridaní validačných atribútov do triedy CubEvent je vlastnosť Title povinný parameter so zobrazovaným názvom Event title, minimálnou dĺžkou 2 znaky a maximálnou dĺžkou 100 znakov. Táto implementácia je vo výpise 5.

```
[Required]
[Display(Name = "Event title")]
[StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.",
    MinimumLength = 2)]
public string Title { get; set; }
```

Výpis 5: Validačné atribúty vlastnosti Title

Model je stavebným kameňom MVC aplikácií. Dáta z modelu je potrebné nejakým spôsobom zobraziť. Na to slúži zobrazovací nástroj Razor, ktorý je popísaný v nasledujúcej kapitole.

5.3 Razor

Razor [11] je zobrazovací nástroj, ktorý nahradil Web Forms ASPX zobrazovací nástroj. V ASP.NET MVC 4 nenájdeme triedy s príponou .aspx, ktoré sú typické pre ASP.NET Web Forms. Razor má novú príponu, a to .cshtml. Ako sama časť prípony naznačuje, Razor je elegantný spôsob vytvárania HTML výstupu pomocou jazyka C#. Tento nástroj poskytuje stručný spôsob ako používať značkovanie a kód súčasne v jednom súbore. Zároveň minimalizuje množstvo znakov potrebných pri vytváraní pohľadovej šablóny, vďaka čomu je písanie kódu rýchlejšie a plynulejšie

5.3.1 Syntax

Hlavným rozpoznávacím prvok Razru je symbol @ . Tento symbol označuje bloky kódu, komentárov, výrazov a vnoreného kódu. Presnejšie povedané znak @ je pokynom pre zobrazovací nástroj, aby začal spracovávať obsah pohľadu ako kód, ktorý musí byť analyzovaný, spracovaný a napokon vykonaný.

5.3.2 Bloky kódu

Ukážka zdrojového kódu číslo 6 znázorňuje zápis bloku kódu. Blok kódu [12] je uzavretý v zložených zátvorkách s prefixom @.

```
@{
    ViewBag.Title = "Edit";
 }
```

Výpis 6: Blok kódu v Razor

5.3.3 Výrazy

Výrazy [12] sú časti kódu, ktoré majú návratovú hodnotu. Výrazy môžu vyvolávať metódy. Taktiež môžu byť použité na získanie hodnoty vlastností. Výrazy môžu byť taktiež vložené priamo medzi výstupný HTML kód ako je vidieť vo výpise 7.

```
<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

Výpis 7: Výrazy v Razor

Štandardne je vyhodnotenie akéhokoľvek Razor výrazu kódované pomocou HTML. To znamená, že akékoľvek špeciálne znaky, ktoré majú význam pre klienta sú vynechané ako súčasť výstupu spracovania. Táto funkcia slúži tiež ako bezpečnostný mechanizmus a bráni náhodnému pridaniu spustiteľného kódu, ktorý by bol zaslaný žiadateľovi. V prípade ak potrebujeme zobrazit' hodnotu vlastnosti objektu alebo metódy priamo, mali by sme použit' Raw, pomocnú rozširujúcu HTML metódu.

5.3.4 Inline kódy

Razor zobrazovací nástroj je dost' chytrý na to, aby vedel posúdit', kedy sa má vykonateľný kus kódu ukončit'. Je to veľmi užitočné, ak sú v pohľadoch využívané inline kódy [12].

```
@if (userIsAuthenticated)
{
    <span>Hello, @username</span>
}
else
{
    <a href='#'>Please login</a>
}
```

Výpis 8: Inline kód v Razor

Inline kódy sú ako všetky kódy spracovávané Razrom uvedené prefixom @, v tomto prípade však v sebe obsahuje kód aj HTML značky. Z výpisu 8 je zjavné, že sa jedná o opak výrazov.

5.3.5 Komentáre

Ako v každom programovacom jazyku aj v Razor kóde sú dostupné komentáre [12]. Sú vhodné hlavne v prípade zmien v kóde, ale ak máme zložitejší blok, ktorý by mohol byť ťažko pochopiteľný, iným než autorovi. Komentáre sú uvedené medzi páry značiek @**@.

```
@* toto je komentar *@
```

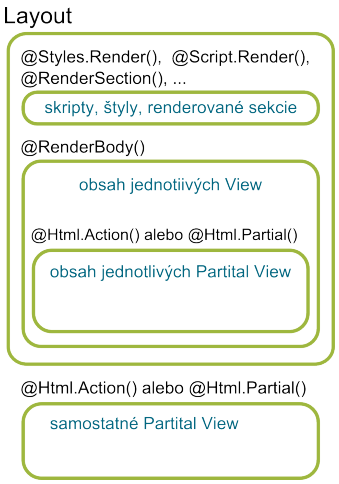
Výpis 9: Komentár v Razor

Pri vytváraní aplikácií spadajúcich pod MVC architektúru, obzvlášť pri vytváraní pohľadov, je dobré držať sa politiky „Don't ask. Don't tell", čiže nezobrazujeme zbytočne informácie navyše. Nasledujúca kapitola podrobne rozoberá pohľady, v ktorých je využívaný práve Razor.

5.4 View

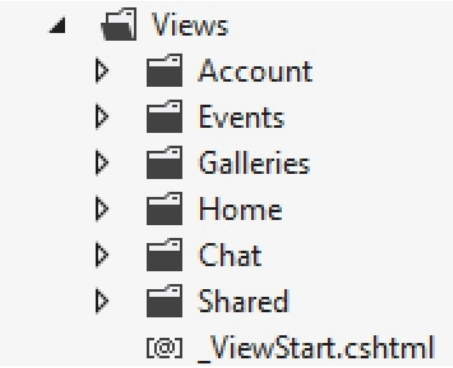
V ASP.NET MVC a v MVC návrhových vzoroch všeobecne, pohľad zabezpečuje prezentáciu dát ako výsledok požiadavku odoslaného kontrolórom. Pohľady poskytujú čistý a dobrý spôsob ako oddeliť záležitosti týkajúce sa prezentácie dát od logiky aplikácie.

Vzhľad každej generovanej stránky je ovplyvnený zvoleným Layoutom (viac v kapitole 5.4.2), obsahujúcim základnú štruktúru stránky, do ktorej sa podľa schémy na obrázku 5.3 vkladajú jednotlivé odkazy na CSS, skripty, alebo samotný CSS, prípadne skriptovací kód. Nasledujú časti pre renderovanie pohľadov a generované sekcie HTML kódu. Čiastočné pohľady (Partial Views) môžu byť buď samostatne, mimo layout alebo vykreslené ako súčasť plnohodnotného pohľadu.



Obrázok 5.3: Zostavenie pohľadu

Štandardne sú pohľady umiestnené vo Views zložke, ktorá je na obrázku číslo 5.4.



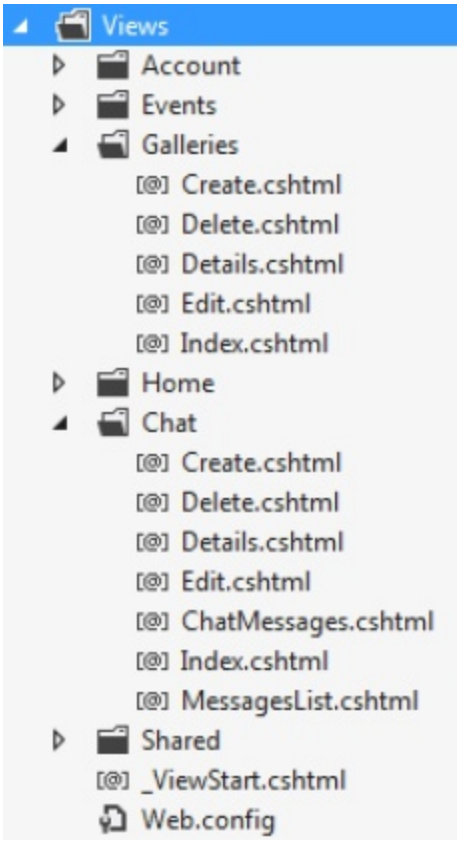
Obrázok 5.4: Zložka Views

V projektovej zložke Views sú ďalšie podzložky, kde každá odpovedá konkrétnemu kontrolórovi, ako je možné vidieť na obrázku 5.5. V týchto podzložkách sa nachádzajú jednotlivé pohľady – súbory s príponou .cshtml, odpovedajúce konkrétnym akciám.

V nasledujúcej kapitole je popísaná zložka Shared, v ktorej sú zdieľané pohľady.

5.4.1 Zdieľané pohľady

Pohľady pre jednotlivé kontrolóri sú umiestnené v zložkách s názvami zhodnými s názvami kontrolórov. Pohľady sú k dispozícii len kontrolórovi, ktorý vlastní danú zložku. Ak chceme zabezpečiť, aby bol pohľad prístupný viacerým kontrolórom, musíme ho označiť ako zdieľaný jeho umiestnením do podzložky Shared v zložke Views. Ak počas behu programu nie je nájdený odpovedajúci pohľad v podzložke pohľadov patriacej da-



Obrázok 5.5: Zložka Views a jej triedy

nému kontrolórovi, je zložka Shared ďalším miestom kde sa pohľad hľadá. Po vytvorení projektu s internetovou aplikáciou sú v zložke Shared primárne tri pohľady.

- **_LoginPartial.cshtml** – ide o čiastočný pohľad, ktorý obstaráva prihlasovaciu kontrolu, táto kontrola je vložená do mnohých pohľadov a zobrazuje prihlasovací formulár pre užívateľa, ktorý nie je autorizovaný
- **Error.cshtml** – je globálny chybová pohľad, ktorý je vrátený metódami, keď sa stane niečo neočakávané
- **NotFound.cshtml** – je pohľad, ktorý je vrátený v prípade, že užívateľ zadá požiadavku na obsah, ktorý nie je možné zobraziť, teda je vrátený status kód 404
- **_Layout.cshtml** – je pohľad, ktorý reprezentuje vzhľadovú šablónu pre každú stránku v aplikácií, obsahuje prvky ktoré sú vždy obsiahnuté a následne v jednotlivých pohľadoch rozšírené

Viac o layoutoch v nasledujúcej kapitole.

5.4.2 Layouts

Layouts (layouty)[13] v ASP.NET MVC sú podobné predlohovým stránkam (master pages) vo Web Forms. Layout je šablóna zabezpečujúca jednotný vzhľad všetkých stránok aplikácie. Layout-y typicky obsahujú skripty, hlavičky, pätičky a iné elementy, ktoré sú považované za dôležité na viac než jednej stránke. Vo chvíli keď sa načíta pohľad, je spracovaný layout a obsah pohľadu je umiestnení na miesto, kde je volanie @RenderBody() metódy.

Je vhodné v aplikáciách používať len jeden layout. Zabezpečí Sa tým jednotný vzhľad stránky. V aplikáciach ASP.NET MVC 4 je tento layout umiestnený v súbore _ViewStart, ktorý je popísaný v kapitole 5.4.3.

5.4.3 _ViewStart súbor

Pri použití rovnakého layout-u vo viacerých pohľadoch je špecifikovanie layout-u v každom pohľade zbytočne opakujúce sa. Na dodržanie konvencie DRY (Don't repeate yourself – neopakuj sa) je vhodné špecifikovať východiskový layout pre všetky pohľady stránky v súbore _ViewStart.cshtml [14]. Tento súbor sa nachádza v zložke Views. Jeho úlohou je nie len špecifikovať layout ale aj uchovávať spoločný kód pre všetky pohľady v aplikácií. Vždy keď je každý pohľad načítaný, je vyvolaný _ViewStart.cshtml. Tento súbor sa správa ako bázová trieda pre všetky pohľady a akýkoľvek kód v ňom je súčasťou konštruktora každého pohľadu na stránke. Ukážkový kód 10 spustí pri začiatku parsovania Razor pohľadu layout _Foundation.cshtml. ktorý zabezpečuje jednotný vzhľad všetkých stránok aplikácie.

```
@{
    Layout = "~/Views/Shared/_Foundation.cshtml";
}
```

Výpis 10: Hlavný Layout

5.4.4 HTML helpers

HTML helpers sú rozširujúce metódy s prefixom Html. Poskytujú rýchly spôsob ako uskutočniť akcie alebo vložiť nejaké informácie do výstupu pohľadu. HTML helpers metódy umožňujú zapuzdrovať funkcionalitu do knižníc na generovanie HTML značiek, ktoré môžu byť následne znovu použité naprieč celou aplikáciou. Typicky je návratovým typom týchto metód reťazec. HTML helpers pomáhajú pri štandardizovaní renderovania obsahu prezentovaného v pohľade ako napríklad formuláre či odkazy. Výpis zdrojového kódu 11 vykreslí textové pole na zadávanie užívateľských vstupov v podobe správy do chatu, stačí pomocou HTML helper metódy vrátiť vstupný element a lambda výrazom určiť aká vlastnosť modelu sa bude nastavovať.

```
@Html.EditorFor(model => model.message)
```

Výpis 11: HTML helper

Na zobrazovanie hodnôt objektov slúži metóda Html.DisplayFor, ktorá je popísaná v nasledujúcej kapitole.

5.4.5 Html.DisplayFor

DisplayFor je rozšírenie základnej verzie metódy Display. DisplayFor je metóda typicky používaná na zobrazovanie hodnôt z objektov, ktoré sú vystavené vlastnosťami modelu. Táto metóda generuje rôzne HTML značky na základe typu dát vlastnosti, ktorá bola renderovaná a podľa toho, či je vlastnosť označená určitými atribútmi.

Metóda vykresľuje značky na základe nasledujúcich pravidiel:

- Ak je vlastnosť typovaná ako primitívny typ (integer, string ...), metóda renderuje reťazec, ktorý reprezentuje hodnotu vlastnosti
- Ak je typ vlastnosti Boolean, metóda renderuje HTML výstupný element zaškrtávacie políčko (check box)
- Ak je vlastnosť označená dátovým typom vlastnosti, atribút špecifikuje značku, ktorá je generovaná pre vlastnosť, napríklad ak je vlastnosť označená atribútom EmailAddress, metóda generuje značky, ktoré obsahujú HTML ukotvenie, ktoré je konfigurované protokolom mailto
- Ak objekt obsahuje viac vlastností metóda vygeneruje pre každú vlastnosť reťazec, ktorý obsahuje značku pre meno vlastnosti a značku pre hodnotu vlastnosti

5.4.6 ViewBag

ViewBag [15] je vlastnosť triedy ControllerBase, z ktorej dedia všetky kontrolóri. Je to dynamický objekt, ktorý využíva dynamické prednosti jazyka C# 4. Namiesto triedenia položiek do slovníka používajúceho reťazové kľúče, stačí nastaviť vlastnosti dynamickej vlastnosti v kontrolórovi. ViewBag slúži na prenos dát z kontrolóra do pohľadu. ViewBag je k dispozícií aj v pohľade. Výpis 12 z kontrolóra vyberie všetky fotografie z jednej galérie a následne ich pomocou vlastnosti ViewBag predá do PicInGallery.

```
public ActionResult Details(int id = 0)
        {
            Gallery gallery = db.GalleriesSet.Find(id);
            if (gallery == null)
            {
                return HttpNotFound();
            }
```

```
            var picInGal = db.PicsGallerySet.Where(PicsGallery =>
                                PicsGallery.galleryID == gallery.galleryID)
                                        .Select(PicsGallery => PicsGallery.eventPic);

            ViewBag.PicInGallery = picInGal;

            return View(gallery);

        }
```

Výpis 12: ViewBag v kontrolórovi

ViewBag v pohľade vo výpise 13 umožňuje hromadné nahrávanie fotiek pomocou cyklu foreach. V tomto cykle sa každá fotka uloží pomocou ViewBag práve do kolekcie PicInGalallery, ktorá bola vytvorená v kontrolórovi.

```
@foreach (var item in ViewBag.PicInGallery)
{
    <div class="picturebox">
        <a href ="/PicturesUpload/@(item.path)" class="galleryImg">
            <img src ="/PicturesUpload/t@(item.path)" class="galleryThumb" />
        </a>
        @Html.ActionLink("x", "PicDelete", new { p_id = item.eventPicID, g_id = Model.galleryID },
              new {@class="DeleteImg" })
    </div>
}
```

Výpis 13: ViewBag v pohľade

Na to aby bolo vôbec možné vlastnosti z modelu priamo použit'v pohľade, je potrebné pohľad špecifikovat'ako silne typový. Viac o silne typových pohľadoch v kapitole 5.4.7.

5.4.7 Silne typové pohľady

Silne typový pohľad spája modelový pohľad so zobrazovacím pohľadom. Jedná sa o pohľad, ktorý prijíma známy typ modelového objektu ako parameter z kontrolórovej akcie metódy. Pohľad je silne typový ak je označený anotáciu @model na začiatku pohľadového súboru. Silne typovaný pohľad vo výpise 14 využíva Gallery model pohľadu.

```
@model IEnumerable<ClubGuide.Models.Gallery>

@{
    ViewBag.Title = "Galleries";
}

<h2>Index</h2>
```

```
<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.addDate)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.galleryTitle)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.addDate)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.galleryTitle)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.galleryID }) |
            @Html.ActionLink("Details", "Details", new { id=item.galleryID }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.galleryID })
        </td>
    </tr>

}
```

Výpis 14: Silne typový pohľad

Silne typové pohľady umožňujú priamy prístup k vlastnostiam modelu. Používanie silne typových pohľadov má niekoľko výhod:

- Vytvorenie pohľadu s <*form*> značkou, ktorá obsahuje všetky polia a validačné možnosti založené na vlastnostiach v pohľade modelu
- Možnosť definovať layout, ktorý slúži ako hlavné rozmiestnenie s elementami ako napríklad logo, horné menu, pätička, to všetko bude rovnaké pre všetky pohľady využívajúce spoločné rozloženie zobrazovaného obsahu. Umožňuje tak zachovať konzistentný vzhľad. Taktiež zmeny vykonané v zdieľanom layoute sú aplikované na všetky pohľady, využívajúce toto rozmiestnenie
- Umožňuje definovať pohľad ako čiastočný. Čiže využiť len časť HTML, ktorá môže byť použitá v iných pohľadoch

Na to aby mohli byť hodnoty vlastností pohľadov zobrazené v pohľadoch, je potrebné tieto dve časti spojiť. Na to slúži kontrolór, ktorý je prostredník v komunikácií pohľadu a modelu. Nasledujúca kapitola podrobne popisuje funkčnosť kontrolóru a všetkých jeho súčastí.

5.5 Controller

Kontrolór spracováva prichádzajúce HTTP požiadavky od užívateľov. Tieto požiadavky sú obstarávané špecifickými kontrolórmi. Kontrolór obsahuje metódy, ktoré spracovávajú HTTP požiadavky. Tieto metódy sú nazvané action methods, z dôvodu, že vracajú objekt typu ActionResult. Kontrolór obsahuje niekoľko takýchto metód, ktoré majú prístupové práva nastavené vždy na verejné. Action methods sú ako lepidlo, ktoré spája dáta z modelu s užívateľským rozhraním aplikácie. Podrobnejšie sú action methods popísané v kapitole 5.5.3.

Kontrolór sa skladá z verejnej triedy, ktorá dedí z bázovej triedy Controller a osahuje verejné metódy, ktoré definujú akcie. Vo výpise 15 je jedna z metód obsiahnutá v triede GalleriesController, ktorá zabezpečuje vytvorenie novej galérie pomocou metódy akcie Create. Táto metóda preberá ako parameter objekt typu Gallery. Telo metódy obsahuje podmienku na overovanie výskytu chýb alebo konverzie dát počas viazania modelu. Pokiaľ je podmienka splnená, nasleduje vloženie galérie do databázy, uloženie zmien a presmerovanie na Index metódu. Nakoniec je vrátený pohľad so zoznamom všetkých galérií.

```
public class GalleriesController : Controller
{
    private GlobalDBContext db = new GlobalDBContext();
    public ActionResult Create(Gallery gallery)
      {
          if (ModelState.IsValid)
          {
              gallery.addDate = DateTime.Now;
              db.GalleriesSet.Add(gallery);
              db.SaveChanges();
              return RedirectToAction("Index");
          }

          return View(gallery);
      }
}
```

Výpis 15: Kontrolór galérie GalleryController

Kontrolór nededí len z triedy Controller. Dedí z ďalších dvoch tried. Podrobnejšie sú všetky tieto triedy popísané v nasledujúcej kapitole.

5.5.1 Bázová trieda ControllerBase, Controller a rozhranie IController

Rozhranie IController [16] definuje základný element kontrolóra, tzv. Execute metódu, ktorá prijíma objekt RequestContext. V metóde Execute je možný priamy prístup k HttpContext, Request a Respons objektom. Implementovanie rozhrania IController využitie metódy Execute znemožňuje vytvorenie pohľadu priamo z kontrolóra, čím sa miešajú dokopy prezentačné záležitosti s kontrolórovou logikou. Je to spôsobené možnosťou písania HTML priamo v rámci kontrolóra. Pri reálnom vývoji je nepravdepodobné priame

implementovanie tohto rozhrania z dôvodu, že sa nevyužijú všetky užitoční vlastnosti frameworku. Vo väčšine prípadov, však kontrolór dedí priamo z bázovej triedy ControllerBase alebo z triedy Controller.

Trieda CotrollerBase [17] implementuje rozhranie IController, avšak aj napriek tejto implementácií, trieda využíva mnoho funkcií frameworku. ControllerBase trieda obsahuje vlastnosť ViewData, ktorá je využívaná v kontrolórovy k odovzdávaniu dát do pohľadu. Aj napriek využívaniu niektorých vlastností frameworku, stále nie je možné z ControllerBase triedy renderovať pohľady ani použiť akcie metód. K zaisteniu tejto funkcionality, je potrebné, aby kontrolór dedil z triedy Controller [18].

Táto trieda dedí z triedy ControllerBase, takže obsahuje všetky vlastnosti svojho predka. Oproti IController a ControllerBase má trieda Controller značné množstvo prídavných funkcionalít. Jednou z nich je trieda ControllerActionInvoker, ktorá obstaráva výber konkrétnej metódy, ktorá sa vykoná na základe URL a definuje metódy ako View, ktorá renderuje pohľad priamo z akcie kontrolóra. Pri vytvorení kontrolóra, tento primárne dedí z triedy Controller. Z toho dôvodu nie je potrebné ani moc žiadané aby kontrolór dedil priamo z ControllerBase alebo IController.

Aby bola zabezpečená správna funkčnosť ASP.NET MVC frameworku pri predávaní dát metódam z bázových trieda a rozhrania, je potrebné dáta z databázy vybrať a spracovať. Na to slúži jazyk LINQ, ktorému je venovaná nasledujúca kapitola.

5.5.2 LINQ

Language –Integrated Query [19] je dotazovací jazyk známy pod skráteným názvom LINQ. Tento jazyk je uľahčením práce so zdrojmi dát, nad ktorými je potrebné vykonávať dotazy. Pred existenciu LINQu musel programátor vedieť špecifický jazyk, pracujúci s konkrétnym druhom dát ako napríklad SQL pre prácu s relačnými databázami alebo XQuery pre prácu s XML. LINQ poskytuje jednotný spôsob zápisu dotazu nad dátovými zdrojmi implementujúcimi generické rozhrania IEnumerable alebo IQueryable. V samotnom dotaze sú špecifikované práve tie informácie, ktoré majú byť načítané zo zdroja dát. V dotaze je možné špecifikovať aj spôsob, akým majú byť dáta zoradené, zoskupené a v akom tvare, pred tým než sú vrátené. V LINQu sú dotazy ukladané do premennej s názvom var. Táto premenná nevracai žiadne dáta, slúži výlučne na ukladanie informácií dotazu. Po vytvorení dotazu je potrebné dotaz vykonať, aby vybrané nejaké dáta.

Syntax založená na výrazoch umožňuje písanie dotazov na vysokej úrovni. Formátovanie dotazov je podobné tým z T-SQL. Používaním tohto spôsobu zápisu je možné komplexnejšie filtrovanie, zoraďovanie a zoskupovanie operácií vykonávaných nad dátovými zdrojmi s použitím minimálneho kódu. Pomocou výpisu 16 je možné vybrať všetky záznamy z tabuľky Event pomocou jednoduchého dotazu na inštanciu ClubEventSet. Dotaz sa skladá z premennej var, do ktorej sa uloží výsledok dotazu. Následne sa určí odkiaľ sa budú brať dáta a v poslednom riadku sa vyberú spracované dáta, ktoré vyhovujú dotazu. Samotný výber dát prebieha ako posledný, kedy je už jasné, ktoré dáta budú uložené do premennej var. Samotný dotaz je veľmi podobný cyklu froeach. V porovnaní s dotazmi používanými napríklad pri príci s T-SQL databázami je zápis opačný.

```
var events = from e in db.ClubEventsSet
                        select e;
```

Výpis 16: LINQ dotaz

Dotazy založené na metódach, využívajú postupnosť volaní metód, využívajúce lambda výrazy ako parametre. Tento spôsob zápisu je o niečo zdĺhavejší než predošlý a nie je moc preferovaný na prácu s databázami.

Dáta vybrané pomocou LINQ dotazov sú najčastejšie predávané a ďalej spracovávané pomocou action methods v nasledujúcej kapitole.

5.5.3 Action methods

V ASP.NET MVC je špecifický typ metódy nazvaný akcia metódy [20] alebo skrátene akcia. Kód obsiahnutý v týchto metódach obstaráva biznis rozhodnutia na základe dát, ktoré im boli predané, interakciu s modelom na základe požiadavky a počas behu programu určujú, ktorý pohľad by mal byť spracovaný v odpovedi pre klienta. Vo väčšine prípadov je návratovým typom akcie metódy objekt ActionResult, ale nemusí tomu byť tak vždy. Je možné použiť aj iné návratové typy ako void, string a podobne. V prípade, že metóda vracia pohľad, súbor, alebo presmerovanie na iný pohľad, je návratovým typom ActionResult. Táto trieda je bázovou triedou pre všetky svoje špecifické implementácie.

Nasledujúce triedy dedia z triedy ActionRestults:

- **ViewResult** – vracia pohľad, ktorý renderuje HTML v prehliadači, je to jedna z najpoužívanejších implementácií ActionResult
- **PartialViewResult** – na rozdiel od ViewResult vracia len čiastočný pohľad
- **ContentResult** – vracia akýkoľvek typ obsahu, primárne sa používa na vrátenie holého textu, ale je možné explicitne určiť typ obsahu (napr. text/xml)
- **EmptyResult** – je ekvivalentom k metóde void, nevracia žiaden obsah
- **FileResult** – vracia binárny obsah (napr. súbor na stiahnutie)
- **HttpUnauthorizedResult** – používa sa v prípade, že sa požiadavka snaží pristúpiť k vyhradenému obsahu, napríklad obsah vyhradený len pre prihláseného užívateľa
- **JavaScriptResult** – vracia JavaScript kód
- **JsonResult** – vracia objekt vo formáte JavaScript Object Notation (JSON)
- **RedirectResult** – používa sa na HTTP presmerovanie na inú URL
- **RedirectToRouteResult** - slúži na http presmerovania na konkrétnej route ceste, nie na URL

Vo výpise 17 je zdrojový kód metódy Index s návratovým typom ActionResult, ktorá vráti pohľad so zoznamom udalostí.

```
public ActionResult Index()
        {
            return View(db.ClubEventsSet.ToList());
        }
```

Výpis 17: Metóda Index s návratovým typom ActionResult

Trieda FileResult je podrobnejšie popísaná v nasledujúcej kapitole z dôvodu jej využitia v ukážkovej aplikácií na sťahovanie kalendára udalostí.

5.5.4 FileResult

FileResult je abstraktná bázová trieda dediaca z ActionResult, ktorá umožňuje posielanie binárnych súborov v odpovedi. Z tejto triedy dedí hneď niekoľko tried, obstarávajúcich stiahnutie súboru. Medzi základné patria:

- **FileContentResult** – trieda je použitá pri vrátení bytového poľa v podobe súboru
- **FilePathResult** – v prípade vrátenia súboru, ktorý je uložený na disku, sa použije trieda FilePathResult, ktorej sa ako jeden z parametrov predá cesta k súboru
- **FileStreamResult** – táto trieda vráti obsah otvoreného streamu v podobe súboru

Ukážková metóda DownloadFile vo výpise 18 s návratovým typom FileResult, do parametru *id* predá *id* konkrétnej udalosti. Pomocou inštancie StringBuilder [21] sa do súboru zapíšu požadované dáta, podľa špecifikácie súborov typu iCalendar [22]. Následne je do bytového poľa uložený text zostavený StringBuilderom, ktorý je zakódovaný pomocou UTF-8 a prevedený do reťazca. Obsah bytového poľa je pomocou inštancie FileContentResult uložené do súboru reprezentujúceho iCalendar. Vlastnosť FileDownloadName umožní nastavenie mena súboru spolu s odpovedajúcou príponou. Na záver metódy je vrátený obsah novovytvoreného súboru FileContentResult v podobe objektu result.

```
public FileResult  DownLoadFile(int id)
        {
            ClubEvent clubevent = db.ClubEventsSet.Find(id);
            StringBuilder  sb = new StringBuilder();
            sb.AppendLine("BEGIN:VCALENDAR");
             ...

            Byte[]  bytes = Encoding.UTF8.GetBytes(sb.ToString());
            FileContentResult result  = new FileContentResult(bytes, "text/calendar");
            result.FileDownloadName = "event.ics";
            return result;
        }
```

Výpis 18: Metóda DownloadFile

V nasledujúcich dvoch kapitolách je popísané ošetrenie neplatnej požiadavky na vrátenie obsahu. Tento prípad môže nastať aj pri sťahovaní súborov, preto je potrebné ošetriť tieto prípady.

5.5.5 HttpNotFoundResult

HttpNotFoundResult je trieda, ktorá dedí z HttpStatusCodeResult. Táto trieda obstaráva prípady, kedy užívateľ požaduje obsah, ktorý nie je dostupný. Pomocou metódy HttpNotFound je možné zobraziť užívateľovi po zadaní nesprávnej požiadavky alebo v prípade podvrhnutia ID požiadavky, chybovú stránku s http status kódom 404 – stránka nebola nájdená. Avšak táto metóda je značne obmedzená. Neposkytuje mechanizmus na zobrazenie prispôsobenej chybovej stránky. Jediné prispôsobenie je v podobe definovania vlastnej chybovej správy, preto ak chceme zobraziť vlastnú chybovú stránku je potrebné využiť triedu NotFoundResult z nasledujúcej kapitoly.

5.5.6 NotFoundResult

Trieda NotFoundResult z výpisu 19 dedí priamo z triedy ActionResult, ktorá poskytuje implementáciu metódy ExecuteResult. Táto metóda nastavuje status kód odpovede na 404 a nastavuje názov pohľadu, ktorý sa bude renderovať. Týmto pohľadom je pohľad NotFound.

```
public class NotFoundResult : ActionResult
    {
        public override void ExecuteResult(ControllerContext context)
        {
            context.HttpContext.Response.StatusCode = 404;
            new ViewResult { ViewName = "NotFound" }.ExecuteResult(context);
        }
    }
```

Výpis 19: Trieda NotFoundResutl

Metóda NotFound vo výpise 20 je definovaná v kontrolórovi ErrorController. Táto má návratový typ ActionResult a vracia novú inštanciu triedy NotFoundResult. Trieda NotFoundResult umožňuje vytvorenie vlastnej chybovej stránky.

```
public ActionResult NotFound()
        {
            return new NotFoundResult();
        }
```

Výpis 20: Metóda NotFound

Aby mohla byť chybová stránka zobrazená je potrebné nastaviť vo WebConfigu element customerrors, ktorý umožňuje vlastné spracovanie chýb. Nastavenie tohto elementu je vo výpise 21.

```
<customErrors mode="On" defaultRedirect="/Error/NotFound/">
</customErrors>
```

Výpis 21: Nastavenie customErrors

6 Ajax a jQuery

Súhrnné označenie technológií pre vývoj interaktívnych webových aplikácií Ajax [23], umožňuje menit' obsah stránok, bez potreby znovu načítania celých stránok zo serveru. Toto prináša výhodu v podobe rýchlejšieho načítania obsahu stránky užívateľovi a zníženie záťaže serveru, ktorý generuje len požadovanú časť stránky, nie celý jej obsah. Ajax je skratka pre Asynchronous JavaScript a XML. Ajax umožňuje vytvárat' požiadavky na server na pozadí z kódu na klientskej strane. U Ajax-u je podstatné to, že je asynchrónny. Je možné vytvorit' požiadavku a následne spracovat' odpoveď, kedykoľvek je pripravená na spracovanie udalosťou, ktorá je vyvolaná, keď je požiadavka kompletná.

ASP.NET MVC využíva koncept Unobtrusive JavaScript [5], ktorý oddeľuje JavaScript kód od HTML kódu. Implementácia funkcionality je napísaná v oddelených súboroch, ktoré sú odkazované v stránke. Unobtrusive JavaScript, alebo tiež nevtieravý JavaScript je spôsob zápisu JavaScriptu tak, aby bola každá stránka plnohodnotne načítaná aj v prípade, že prehliadač nepodporuje JavaScript alebo ho užívateľ zámerne blokuje. Unobtrusive JavaScript je založený na knižnici jQuery JavaScript.

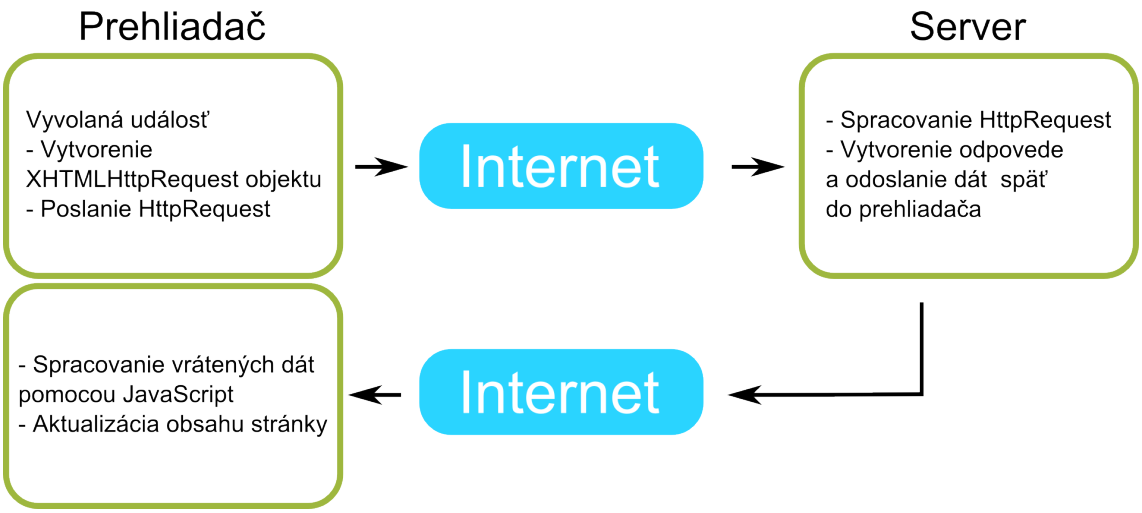
Funkcionalita jQuery [23] spočíva vo vrstve abstrahujúcej prácu jednotlivých prehliadačov s Document Object Model (DOM). Vývojárom tak umožňuje mat' zjednotený model, ktorý eliminuje rozdiely naprieč prehliadačmi. jQuery zabezpečuje mapovanie JavaScriptu a HTML elementov do implementácie jednotlivých prehliadačov.

6.1 Ajax

Ajax je technológia, ktorá zmenila spôsob akým webové stránky odpovedajú na užívateľské vstupy. Ajax predstavil nový vzor akým sa aktualizuje obsah stránky. Používanie Ajaxu na obnovovanie obsahu stránky má niekoľko výhod:

- Požiadavky sú asynchrónne, umožňujú tak užívateľovi plnohodnotné používanie stránky
- Predloha stránky je načítaná len raz, pri prvej požiadavke
- Odpovede serveru sú malé, obsahujú len potrebné informácie, ktoré boli požadované, nie plne novú stránku
- Aktualizácia obsahu stránky môže prebiehat' len v oblasti, kde je táto aktualizácia potrebná

Jazyk na strane serveru spracováva užívateľskú požiadavku a vracia výsledok späť užívateľovi. Všetko sa deje na pozadí, pomocou JavaScript XMLHttpRequest [24], ktorý sa používa na odosielanie HTTP alebo HTTPS požiadaviek na server a k spätnému načítaniu dát z odpovedi serveru do skriptu. JavaScript zohráva hlavnú úlohu pri zväzovaní týchto technológií dohromady a vytvára asynchrónne interakcie medzi serverom a klientom. Popis tejto interakcie je zobrazený na obrázku 6.1.



Obrázok 6.1: Princíp fungovania Ajax

6.2 jQuery

Knižnice jQuery sú jedny z najrozšírenejších JavaSript knižníc. Knižnice jQuery sa nachádzajú v zložke Scripts. Knižnice majú vždy názov vo formáte jquery-[verzia].js. K prístupu a práci s knižnicou je potrebné obsiahnuť daný súbor knižnice v stránke, ktorá ju bude používať. Existujú dva spôsoby ako obsiahnuť knižnicu v danej stránke. Prvým je pridanie referencie na súbor jQuery v stránke pomocou značky <script> . Tento spôsob pridania je jednoduchý. Stačí pridať nasledujúce značky do hlavičkovej značky stránky ako je tomu vo výpise 22.

```
<script src="~/Scripts/jquery-1.9.1.min.js"></script>
```

Výpis 22: Pridanie knižnice jQuery

Druhým spôsobom je zahrnutie súboru do bandle, čo môžeme prirovnať k akémusi balíčku. Bundling (zväzovanie) v ASP.NET umožňuje kombinovať viacero súborov do jedného. Takéto pomyselné zabalenie redukuje veľkosti súborov, nie len JavaScriptu ale aj CSS a obrázkov. Zredukovanie veľkosti týchto súborov je významné z hľadiska množstva odoslaných http požiadaviek, času načítania stránky a celkového množstva prenesených dát. Na zabalenie kódu do týchto balíčkov sa používajú triedy ScriptBundle a StyleBundle. Tieto triedy obsahujú metódu Include(), ktorá umožňuje pridať súbor do balíčka. Tieto balíčky sú pridávané do triedy BundleConfig umiestnenej v zložke App_Start. Trieda obsahuje statickú metódu RegisterBundles(). Táto metóda prijíma ako parameter BundleCollection typ, do ktorého sa pridávajú balíčky. Aby bolo možné tieto baličky využiť v danej stránke, je potrebné použiť triedy ako vo výpise 23.

```
@Scripts.Render("~/bundles/jquery")
@Styles.Render("~/Content/foundation/css")
```

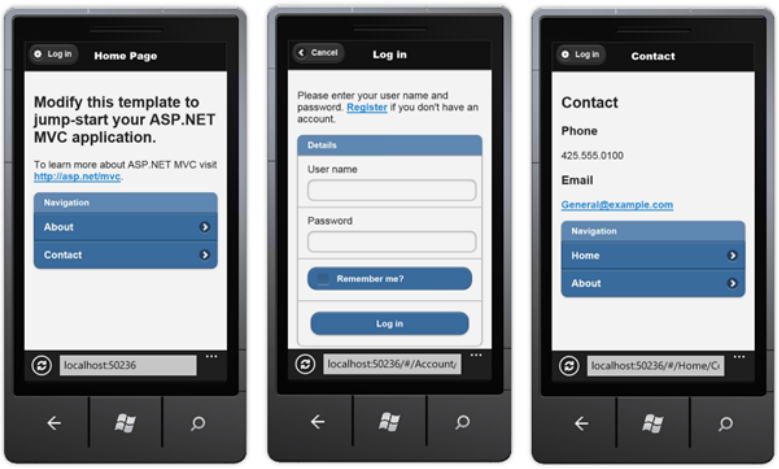
Výpis 23: Bundling knižnice jQuery

7 Testovanie aplikácie

Testovanie ukážkovej aplikácie prebiehalo na localhoste na servery IIS Express. Testovanie prebiehalo na fyzickom zariadení s platformami Windows phone a prostredníctvom simulovaných testov vo Visual Studiu. Cieľom testov bolo odhalit'prípadné nedostatky a chyby. Prvým krokom testovania bolo zobrazenie na emulátore Windows phone v nasledujúcej kapitole.

7.1 Mobilný emulátor

K zaisteniu správnej funkčnosti aplikácie je potrebné dôkladné testovanie pre všetky dostupné mobilné zariadenia a ich internetové prehliadače. Takýchto zariadení sú rádovo stovky tisíc a teda nie je fyzicky možné ich otestovat' všetky. Na to slúžia emulátory internetových prehliadačov pre mobilné zariadenia ako je vidiet'na obrázku 7.1. V konfigurácií každého emulátoru je možné nastavit'typ zariadenia, na ktorom chceme aplikáciu testovat'. Dôležitým aspektom je hlavne rozlíšenie displeja. Každé zariadenie ho má iné. Zatiaľ čo iPhone 5 má rozlíšenie obrazovky 1136 na 640 pixelov, Samsung Galaxy S3 má 1280 na 720 pixelov. Vývojové prostredie Visual Studio 2012 podporuje funkciu multiple browsers (zobrazovanie na viacerých prehliadačoch zároveň). Takýmto spôsobom môžeme aplikáciu jediným spustením zobrazit'na všetkých nami vybraných prehliadačoch a emulátoroch.



Obrázok 7.1: Emulátor

Už pri testovaní pomocou emulátoru som pozorovala určité nedostatky hlavne pri zobrazovaní jednotlivých prvkov stránok. V nasledujúcej kapitole je tento problém podrobnejšie popísaný.

7.2 Vzhľad aplikácie

Po vytvorení akéhokoľvek web projektu vo Visual Studiu 2012 (okrem prázdneho projektu), je primárne užívateľské rozhranie - vzhľad celej aplikácie generovaný pomocou jQuery UI [25]. Jedná sa o framework užívateľského rozhrania postaveného na knižnici jQuery. Tento framework využíva vlastné preddefinované kaskádové štýly. Celkový východiskový vzhľad stránok pôsobí kompaktne a uhladene. Farebná kombinácia je príjemná na pohľad, nijako neruší ani neodvádza pozornosť od obsahu. Výsledná aplikácia využívajúca tento framework je taktiež responzína. Problém tohto frameworku nastáva pri škálovaní stránky. Počas zväčšovania alebo zmenšovania okna prehliadača je možné pozorovať nepríjemné preblikávanie jednotlivých komponent stránky. Tento efekt nie je žiadúci. jQuery UI je dosť pomalé pri vykresľovaní jednotlivých komponent, z dôvodu veľkého množstva jQuery/JavaScript kódu na pozadí.

Z týchto dôvodov som sa rozhodla vymeniť jQuery UI framework za open source framework Foundation spoločnosti Zurb [26]. Vďaka tomu, že framework je open source je možné ho upravovať podľa vlastných požiadaviek. Najväčšou prednosťou tohto frameworku je sadzba formulárov prispôsobených pre mobilné zariadenia a menšie množstvo jQuery/JavScript kódu na pozadí.

7.3 iCalendar

V ukážkovej aplikácií je možné hneď z úvodnej stránky prejsť na detail akcie po kliknutí na jej popis a následne si stiahnuť súbor kalendára s danou akciou. Tento súbor je typu .ics - iCalendar. Jedná sa o štandard pre výmenu kalendárových dát. Formát súboru je podporovaný všetkými hlavnými mobilnými platformami. Aj napriek tomu, že sa jedná o štandard, každá platforma pracuje so súborom inak. Zatiaľ čo iOS a Android 4.* súbor automaticky integrujú priamo do kalendára zariadenia, Windows phone a Andrioid staršej verzie než 4 vyžadujú na spracovanie súboru nainštalovanie externej aplikácie. Tento nedostatok by mal byť odstránený vo verzií Windows Phone 8.1.

7.4 Testovanie na fyzikom zariadení

Testovanie na fyzickom zariadení bolo zamerané na celkovú funkčnosť a užívateľskú príveticosť. Užívateľ postupne vyskúšal všetky možnosti aplikácie ako prezeranie všetkých stránok, registráciu, prihlásenie a odhlásenie, stiahnutie kalendára, prechádzanie obrázkov galérie v plnom zobrazení a písanie správ do chatu. Testovanie prebiehalo na chytrom mobilnom telefóne Nokia Lumia 620 s operačným systémom Windows Phone 8 a rozlíšením displeja 800x480 pixelov. Testovací užívateľ hodnotil svižnosť stránky veľmi dobre. Načítanie jednotlivých stránok bolo rýchle. Menu je intuitívne a prehľadné. Registrácia nového užívateľa je jednoduchá a ľahká. V prípade chybne zadaného alebo nezadaného povinného parametra sa zobrazí jasná a zrozumiteľná chybová hláška. Pri prehliadaní galérie a otáčaní displeja je otočenie plynulé, bez zásekov. Užívateľ zhodnotil galériu ako veľmi dobrú v porovnaní sinými, ktoré doteraz vyskúšal. Taktiež otáčanie stránok je dobre prispôsobené tomu, aby sa všetky prvky prispôsobili rozmerom a orientácií displeja.

Testovanie neprebiehalo len na fyzickom zariadení ale aj prostredníctvom simulovaných testov na počítači, ktorého špecifikácia je ďalšej kapitole.

7.5 Testovací server

Testovacím serverom bol lokálny počítač s parametrami:

- Procesor AMD Turion 2 s 2,49 GHz
- Pamät 4GB DDR3
- Operačný systém Windows 8.1 64bit

Na tomto počítači prebiehali testy z nasledujúcej kapitoly.

7.5.1 Simulované testy

Vo vývojom prostredí Visual Studio 2012 Ultimate je možné vytvoriť niekoľko druhov simulovaných testov. Testy sú vytvorené ako nový projekt. Na testovanie ukážkovej aplikácie bol vytvorený jeden Unit Test, ktorý obsahoval jeden záťažový test(Load Test) a tri výkonnostné testy (Web Performance Test) Simulovaný záťažový test (Load Test) trval jednu hodinu a prebiehal na simulovanom smartphone. Obsahoval výkonnostné testy zamerané na registráciu užívateľa, odosielanie správ v chate, prezeranie galérie, načítanie domovskej stránky. Maximálny prístup bol obmedzený na 120 užívateľov. Test galérie mal nastavenú periódu päťdesiat spustení na jedného užívateľa za hodinu. Správy v chate boli testované v perióde štyridsať správ za hodinu na užívateľa. Posledný výkonnostný test bol zameraný na registráciu užívateľa a následné automatické prihlásenie. Tento test virtuálny užívateľ vykonal desaťkrát za hodinu.

7.5.2 Výsledky testu

Po ukončení testu sa zobrazia výsledky, kde je možné nájsť informácie o priebehu čiastkových testov, zoznam piatich najpomalších stránok, vyskytnuté chyby, celkový súhrn testu a podobne.

Dôležité údaje sú obsiahnuté v tabuľke 7.1.

Počet užívateľov	120
Doba trvania (hh:mm)	01:00
Počet zobrazených stránok za sekundu	20
Počet požiadaviek za sekundu	377
Priemerná doba načítania stránky (sek.)	1,9
Priemerná doba odpovedi na požiadavku (sek.)	0.11

Tabulka 7.1: Výsledky testu

Aj napriek dobrým celkovým výsledkom testu, nastali počas jeho behu problém týkajúce sa nedostatočného výkonu procesoru. Problémy nastali pri pripojení viac než päťdesiatich užívateľov. Procesor bol aplikáciou vyťažený na deväťdesiatosem percent, procesorový čas bol nedostatočný a odozvy jednotlivých stránok sa zvýšili. Z toho vyplýva, že na reálne prevádzkovanie aplikácie je potrebný výkonnejší hardware.

8 Záver

Cieľom tejto bakalárskej práce bolo oboznámit' sa s technológiou ASP.NET MVC 4 a na praktickej ukážke demonštrovat' využitie tejto technológie so zameraním na stále viac rozšírené chytré mobilné telefóny a tablety.

Hlavným zdrojom informácií a postupov pri vývoji aplikácie bola odborná literatúra a online zdroje. Práca obsahuje všetky body zadania.

V prípade reálneho nasadenia ukážkovej aplikácie, by bolo vhodné dlhšie testovanie hlavne na reálnych užívateľoch a následné zhodnotenie spätnej väzby. Aplikácia bola vytvorená so zámerom používania bežnými užívateľmi. Preto je ich hodnotenie dôležitejšie než výsledky simulovaných testov.

Počas písania tejto práce spoločnost' Microsoft vydala dve nové verzie frameworku ASP.NET MVC. Pre čo najlepšie využitie dostupných technológií by bolo vhodné zvážit' prechod na novšiu verziu frameworku s radou noviniek, ktoré môžu prispiet' k lepšiemu výkonu aplikácie.

9 Literatúra

[1] A. Murin, "Kompatibilita web stránok s mobilnými zariadeniami - prečo a ako." http://www.riant.sk/blog/kompatibilita-web-stranok-s-mobilnymi-zariadeniami-preco-a-ako.

[2] Microsoft, "Asp.net mvc." http://www.asp.net/mvc.

[3] Microsoft, "Using templated helpers to display data in asp.net mvc." http://msdn.microsoft.com/en-us/library/ee308450(v=vs.100).aspx.

[4] Microsoft, "Performing simple validation." http://www.asp.net/mvc/tutorials/older-versions/models-(data)/performing-simple-validation-cs.

[5] W. W. W. Consortium, "The principles of unobtrusive javascript." http://www.w3.org/wiki/The_principles_of_unobtrusive_JavaScript.

[6] Bootstrap, "Bootstrap front-end framework." http://getbootstrap.com.

[7] w3schools, "Asp.net mvc intro." http://www.w3schools.com/aspnet/mvc_intro.asp.

[8] Microsoft, "Formsauthentication.setauthcookie method." http://msdn.microsoft.com/en-us/library/system.web.security.formsauthentication.setauthcookie(v=vs.110).aspx.

[9] A. Meadows, *ASP.NET MVC 4 Mobile App Development*. Packt publishing, 2013.

[10] R. Anderson, "Adding validation to the model." http://www.asp.net/mvc/tutorials/mvc-4/getting-started-with-aspnet-mvc4/adding-validation-to-the-model.

[11] T. FitzMacken, "Razor view engine." http://www.asp.net/web-pages/tutorials/basics/2-introduction-to-asp-net-web-programming-using-the-razor-syntax.

[12] S. Guthrie, "Introducing "razor" – a new view engine for asp.net." http://weblogs.asp.net/scottgu/archive/2010/07/02/introducing-razor.aspx.

[13] S. S. Shekhawat, "Asp.net mvc 4 - layout and section in razor." url http://www.c-sharpcorner.com/UploadFile/3d39b4/Asp-Net-mvc-4-layout-and-section-in-razor/.

[14] E. H. M. H. J. S. Jeffrey Palermo, Jimmy Bogard, *ASP.NET MVC 4 in Action*. Manning Publications, 2012.

[15] M. Habib, "What is viewdata, viewbag and tempdata?." http://www.codeproject.com/Articles/476967/WhatplusisplusViewData-cplusViewBagplusandplusTem.

[16] Microsoft, "Icontroller interface." http://msdn.microsoft.com/en-us/library/system.web.mvc.icontroller(v=vs.118).aspx.

[17] Microsoft, "Controllerbase class." http://msdn.microsoft.com/en-us/library/system.web.mvc.controllerbase(v=vs.118).aspx.

[18] Microsoft, "Controller class." http://msdn.microsoft.com/en-us/library/system.web.mvc.controller(v=vs.100).aspx.

[19] Microsoft, "Linq (language/integrated query)." http://msdn.microsoft.com/en-us/library/bb397926.aspx.

[20] Microsoft, "Controllers and action methods in asp.net mvc applications." http://msdn.microsoft.com/en-us/library/dd410269(v=vs.100).aspx.

[21] Microsoft, "Stringbuilder class." http://msdn.microsoft.com/en-us/library/system.text.stringbuilder(v=vs.110).aspx.

[22] D. S. F. Dawson Lotus, "Internet calendaring and scheduling core object specification (icalendar)." http://www.ietf.org/rfc/rfc2445.txt.

[23] J. R. G. Paz, *Beginning ASP.NET MVC 4*. Paul Manning, 2013.

[24] w3schools, "The xmlhttprequest object." http://www.w3schools.com/xml/xml_http.asp.

[25] T. jQuery Foundation, "jquery user interface." http://jqueryui.com.

[26] Zurb, "Zurb framework." http://foundation.zurb.com.

A Obsah CD

Obsah CD Na priloženom CD sú uložené nasledujúce dáta:

- Text práce vo formáte PDF
- Zdrojový kód ukážkovej aplikácie